The Development and Application of HERMES:

A Heuristic Range-finding Motion-analyzing Electronic System

Josh Goldstein, Anand Kapadia, and Alex Xu

TJHSST Automation and Robotics Research Lab

2014

Table of Contents

Abstract …………………………………………………………..……..…. 3

Introduction ……………………………………………………..……. 3

Literature Review ………………………………………………………….. 5

RoboCup and Visual …………………………………………………….. 5

NASA Rover and Visual ………………………………………………… 7

StreetMap and Visual………..……………………………………………… 8

Methodology ………………………………………………………………. 9

Hardware and Flow Chart………………………………………………… 9

Programming and Flow Chart…………………………………………... 10

Results with Bill of Materials and Gantt Chart…….…………………..…….. 15

Discussion ……………………………………………………………….…... 19

Conclusion ……………………………………………………………….… 20

Final Comments ………………………………………………………….. 21

Appendix ………………………………………………………………. 23

Abstract

In this project, we are combatting the self-localization problem, in which a robot comprehends its general global environment but is unaware of its local surroundings when moving towards a specified location. Therefore, such a robot should use sensors to acquire ambient data and respond to changes from the local environment's 'normal state.' Our planned implementation had involved a laser range finder that detected moving objects blocking the robot's path, and adjusted accordingly. This goal was not realized. However, we have achieved our primary objective, as the robot maneuvers autonomously through an environment by the means of a global Cartesian coordinate system. As the robot travels at a given velocity measured with optical encoders and a direction ascertained by a digital compass, a running program constantly produces displacement vectors to calculate the robot's new global coordinates. The robot continues to adapt to its local environment until it arrives at its final destination.

Introduction

Self-localization is gaining relevance in the technological sphere. It carries significance in the field of robotic engineering, as researchers devise algorithms that help robots effectively respond to new and unpredictable situations. Current research and development focus on robots that must maneuver over terrain that is either too dangerous or not feasible for humans. For instance, bomb disposal robots traverse battlegrounds to locate and deactivate improvised explosive devices, decreasing human casualties (UK Ministry of Defense, 2010). Also, NASA employs planetary rovers that capture surface video footage and mineral samples on Mars, on which humans cannot currently land due to fuel constraints (Gerbis, 2012). The application potential of self-localizing robots is rapidly expanding, especially in fields that desire to limit human risk such as the military. These robots can replace humans operating in threatening environments and adapt in a manner comparable to the trial-and-error nature of the human mind.

Currently, the modus operandi of self-localization is the Global Positioning System, more widely known as GPS. GPS determines the location of a signal using satellites in medium Earth orbit and can be effectively used by people to navigate roads. However, remote sensing has its limitations. Firstly, the GPS systems that we use generally are accurate to about 8 meters (GPS Accuracy, 2014). In an environment that requires precise maneuvering, GPS is not a feasible option. Moreover, GPS units demand a constant satellite signal. Thus, GPS cannot be applied indoors for small scale operations, or for rovers that cannot broadcast their location, such as those in military zones.

The Heuristic Expeditionary Range-finding Motion-analyzing Electronic System (HERMES) rover addresses these problems. The goal of the robot is that with input data of the local environment, HERMES can accurately find the shortest path and navigate it without using satellite signals. Due to the nature of its algorithm, HERMES can also adapt quickly to various coordinate maps. The independence from satellite signals allows for a more precise and widely applicable form of self-localization as well. However, system drawbacks include the inability for a robot to predict variables in the environment, such as moving objects. Similarly, the cost of speed and efficiency is such that the rover requires specific pre-inputted data and cannot find its way through an environment without prior knowledge of the coordinate system.

The primary objective of HERMES is to autonomously move throughout Thomas Jefferson High School for Science and Technology. After a human user inputs a room number, HERMES will move to the specified classroom in no more than eight minutes, utilizing a heuristic lowest-cost search algorithm to reach its destination. Also, HERMES should avoid any hallway obstacles, including students, walls and backpacks, among others, and quickly readjust to continue on its directed path.

Literature Review

When researching self-localization projects to aid in the design of HERMES, three major sources were consulted: articles on RoboCup from MIT, visual self-localization from the University of Chicago, and the Mars rover from NASA. Of the three, MIT's RoboCup proved to be the most significant to the construction of our own robot.

RoboCup:

Hajiaghayi, M. T., & Jamzad, M. (2002). Simple, fast and robust self localization in environments similar to the robocup environment. *mit.edu*.

Engineers implementing the self-localization strategy for their robots need to maintain a global context for their rover to navigate a local environment. Hajiaghayi and Jamzad, two students from MIT competing in the RoboCup contest, achieved this through *global* and *local* self-localization. The RoboCup, an international robotic soccer tournament, requires players to interact with their local surroundings in order to score goals. Hajiaghayi and Jamzad's robot estimated relative distances locally, understanding how to react to its immediate environment. For instance, the distance between the ball, the opposing robot and the walls were all calculated relative the the robot itself, a form of local self localization. Next, the robot determined its movement and action based on a global coordinate system, or *global* self localization. This was used for passing, dribbling, and shooting. However, "global self localization requires more time" and “has higher processing costs,” a downside that must be considered. Hajiaghayi and Jamzad’s solution was to employ a vision algorithm, employing *lines* of various colors to mark boundaries and creating a global coordinate system (1-2). This strategy, while novel in its combination of local and global self-localization to create a more holistic image, is limited by the field of view of the robot. To overcome this pitfall, Hajiaghayi and Jamzad proposed in their conclusion that “use of other sensors like...an electronic compass” will result in a more accurate global map (8).

In studying the research of Hajiaghayi and Jamzad, we were able to develop a few key inferences that proved vital to be in the success of our primary goal. For instance, not restricting ourselves to a “vision” algorithm used for the RoboCup, we also adapted the suggestion of a compass to keep track of direction in a global system. The optical encoders were employed for the same reason. In addition, Hajiaghayi and Jamzad’s concern with the high processing cost of global self localization, our topic of research, we acquired mobile vehicle computer with a high level processor for data analysis. Finally, Hajiaghayi and Jamzad also developed a global map, and traversed it using x, y, and theta inputs, effectively creating displacement vectors to calculate the robot’s new coordinates. Due to the efficiency and elegance of this solution, we adapted it to our own project and created a “preprocessed system” which we traversed through vector displacement, using an electronic compass for direction and optical encoders for magnitude (7).

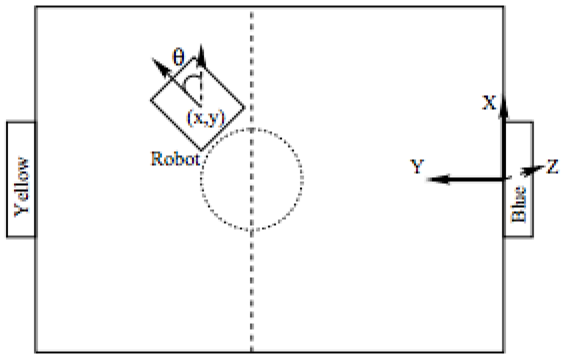


Figure 1. RoboCup robot using its angle in a coordinate plane

NASA Rover:

Olson, C.F. & Matthies, L.H. (1998). Maximum Likelihood Rover Localization by Matching Range Maps. *washington.edu*. Retrieved from http://faculty.washington.edu/cfolson/papers/pdf/icra98.pdf.

Clark Olson and Larry Matthies, researchers from the Jet Propulsion Laboratory at the California Institute of Technology, studied self-localization as an instrument for navigating foreign terrain and applied this concept to the Sojourner Mars rover. This rover overlaid a local map, which was generated by robot cameras, with a global map that was defined by satellite imagery. A program on the rover's on-board computer would then run a probabilistic similarity measure to determine changes in the local environment, adjusting its self-directed movement accordingly. NASA engineers who had controlled previous Mars rover models found that a robot without self-localization could not venture far from the lander and base station, limiting the scientific and exploratory gains made. However, the Sojourner Mars rover and a new Rocky 7 Mars rover prototype, the first extraterrestrial robotic vehicles to implement visual localization, successfully covered more area than their predecessors and developed range maps of their surroundings with stereo cameras. These maps generated three dimensional topographic plots of the surface of Mars, offering scientists information that could only be obtained from the ground.

This article contributed significantly to the HERMES project in that we adopted the idea of overlapping local and global maps as part of our navigation algorithm. The Mars rover

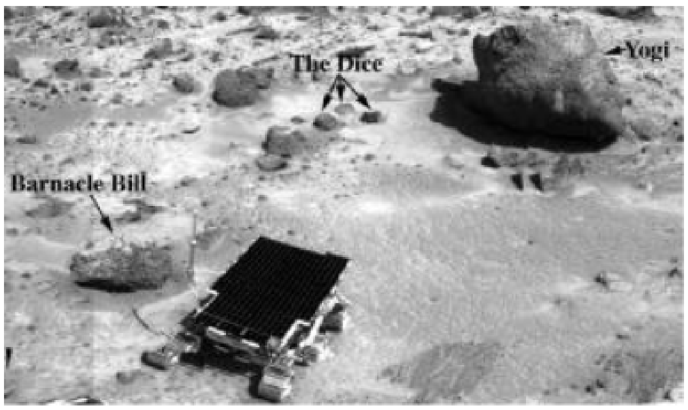


Figure 2. Mars environment

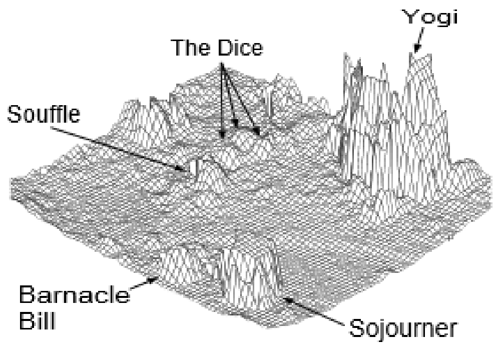


Figure 3. Three-dimensional topographical map

implementation of localization has informed our research and understanding of the problem, as we learned to concentrate on the changes between the local and global images in order to avoid obstacles. The only drawback of this study, as it pertains to our application, is that the researchers used high-precision equipment costing millions of dollars, so their local map was very clearly generated. Our project consisted of greater budget constraints and, as such, we modified the programming subsystem to make up for our limited understanding of the local environment.

StreetMap:

Brubaker, M., Geiger, A., & Urtasun, R. (2012). Lost! leveraging the crowd for probabilistic visual self-localization. *chicago.edu*. Retrieved from http://ttic.uchicago.edu/~rurtasun/publications/brubaker_et_al_cvpr13.pdf.

In the study by Brubaker, Geiger, and Urtasun, a machine was spawned to navigate public roads using online map data and visual odometry, which measures the change in position.

Specifically, the source used was OpenStreetMap (OSM), an open source map platform with community input. The visual odometry system operates cameras and constantly tracks the position of the vehicle on OSM accurate to within three meters. This project is similar to HERMES in that it functions independently of satellite transmission with higher levels of accuracy than GPS. In addition, it can adapt to new situations easily because it uses cameras as its method of self-localization. However, an issue is that the cameras cannot pinpoint a precise location because of many other movements, so the algorithm relies on a probabilistic model and requires complex calculations to determine location. HERMES differs in that it runs without visual odometry or cameras, providing a more exact measurement with less complicated algorithms.

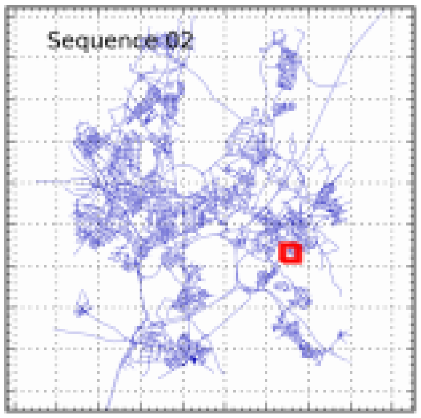


Figure 4. Self-localization on a map created from OpenStreetMap

Methodology

Hardware

The frame for our robot was a DFRobot four-wheel drive mobile platform. Inside was a twelve volt, two ampere battery used to power all the components. We also included an Arduino Leonardo connected to the encoders to calculate rotation of wheels and an Arduino Leonardo with a Monster Moto Shield (MMS) connected to the motors. On top of the robot an Arduino Duemilanove was connected to a compass which was mounted to prevent magnetic interference from the computer that would interfere with the compass readings. The Duemilanove was used instead of the Leonardo because it could handle the stream of compass data. A carputer was anchored on top of the robot and connected to the Arduinos through USB ports. To avoid carrying around wires for the monitor while the robot was moving, remote desktop from a laptop was used to run the programs on the carputer

On the side of the robot, a power switch was attached, linking all parts requiring battery power in a circuit. A 3-D printed cover was added to prevent accidental pressing of the button. Additionally, wires were connected to the leads of the battery and pulled to stick out of the side so that the robot could be charged conveniently. The touch screen and laser range finder, had they been used, would have been powered through the battery directly and the computer respectively. The power switch indicator light was powered through the computer, which was in turn powered by the battery, completing the loop.

Additionally, from the mechanical perspective, a Vex wheel to DC motor adaptor was created. Our original wheels frequently malfunctioned as they were made of foam, and the rubber encasing often slipped off. Therefore, a device was created using set screws to secure the square axle of the vex wheel to the cylindrical axle of the DC motor (see Appendix for a detailed picture).

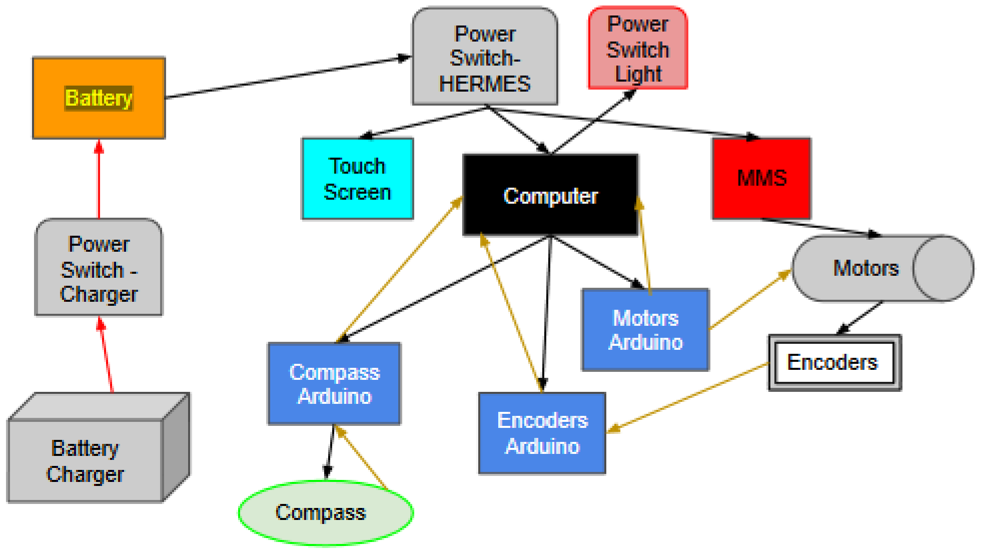


Figure 5. Relationship between elements of HERMES

Program

This project comprised of three Arduino codes and one Python master code. Two Arduino programs would separately read data from the compass and encoders to determine the current direction and heading. The final Arduino program, the motor code, controlled the rotation speed and duration of the motors. The Python code acquired the data from the two input Arduinos, processed the information and subsequently directed the motor code to move the rover.

We retrieved the compass library and default code from the compass that we purchased, the OSEPP Triple-Axis Magnetometer Compass Module. The program assessed the magnetic field strength in the X and Y directions and utilized the arctangent function to find the direction. Compass calibration entailed the measurement of field strengths along each axis, and we adjusted the centroid of compass values to the origin, as indicated in the diagram below.

We added a constant to the resulting compass values in order to normalize them to the local environment, as the earth contains a nonuniform magnetic field that disrupts the standard north-to-south orientation. The code was then uploaded to one Arduino.

The encoder code supported the navigation algorithm by determining the distance traveled. A program outputted the number of rotations of the wheels, reading in from optical encoders that were attached to the motors. We set the rover to travel an exact length and measured the number of wheel rotations in order to learn of the effective distance travelled by one rotation. Additional trials verified the encoder constant, which was multiplied by the raw data values. The encoder code was uploaded to another Arduino.

Figure 6. Compass before calibration (above) and compass after calibration (below)

Dijkstra's algorithm was the means by which to determine the robot's path in the internal coordinate system. Each room contained preset X and Y coordinates that were found by measuring horizontal and vertical distances from a set point designated as the origin in the CAD blueprints of TJHSST. Something important to note was that the rooms were coded as single points in the center of the hallways so that the robot would stay in the center while travelling from room to room. In addition, the hallway intersections were all included so that upon reaching them, the robot could make a decision on whether or not to turn. The graph that would be used for Dijkstra's algorithm contained adjacent rooms for each location. The room data was uploaded in an Excel document, transferred to a text file and read to the Python master code in a dictionary of points. Dijkstra's algorithm was then run to determine the path, using each room as a node.

In the Python code, a robot class included our robot's action methods, such as moveForward(), turnLeft() and turnRight(). The code was programmed to prompt the user to input the current location and the desired location. The code continually received distance and direction calculations from the encoder and compass Arduino codes. Using the data, it determined the correct direction to face and the distance that it needed to travel to arrive at the next room in its path. The robot class methods informed the rover of the desired movements and adjustments. Using this system of continuous updating, the process continued until the robot reached the final destination.

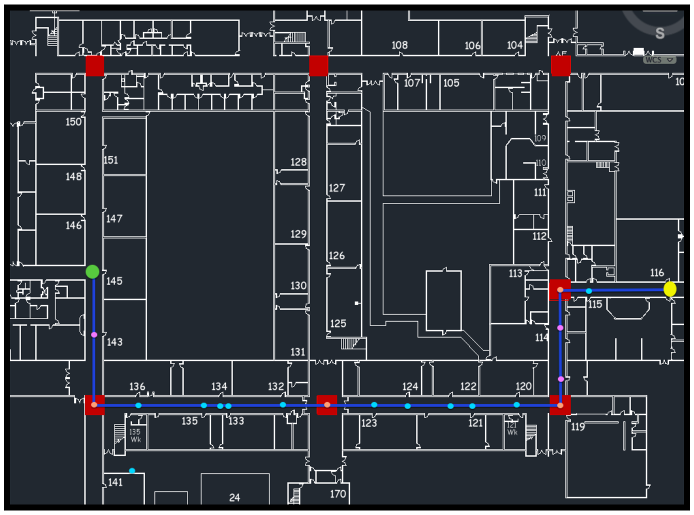


Figure 7. Blueprint of TJHSST used to obtain coordinates

The actual commands for the robot's movement were written in the motor code, which was uploaded onto the third Arduino. The motor code contained the methods to move forward, turn left and turn right.

PySerial dictated all communication between programs. The Arduino input, compass and encoder programs transmitted data to the Python master code, which received and resent the information to the motor code. The input codes sent raw data values through Serial, and the motor code acted upon specific input commands that would signal it to move forward, turn left or turn right at a specified speed.

Originally, a laser range finder (LRF) was intended to appear in the project. It would be able to detect distance from the walls, centering the robot in the hallway. Furthermore, it would recognize obstacles, whether moving or stationary. However, the laser range finder was not included due to time constraints.

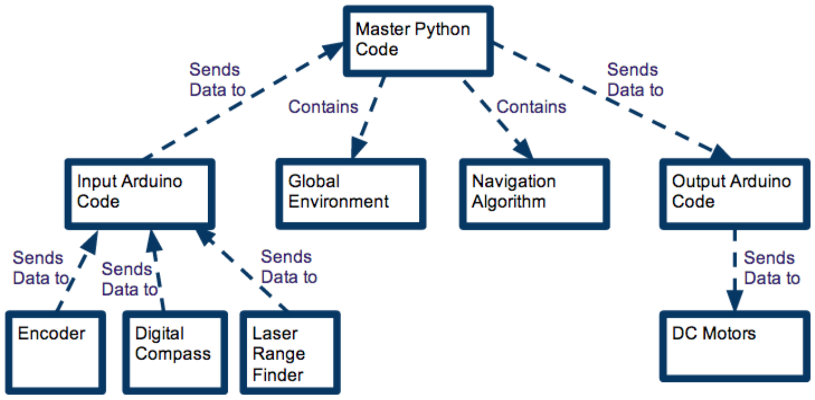
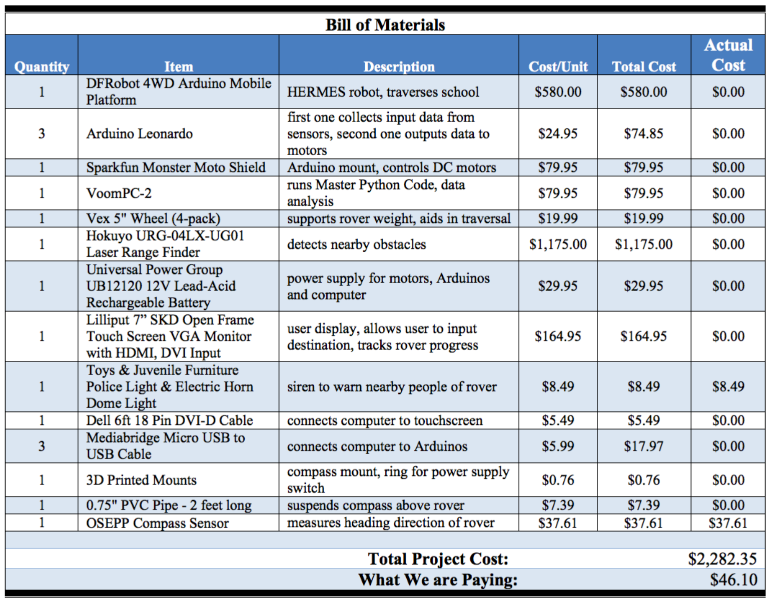


Figure 8. Relationships among codes

Data for the robot's navigation algorithm was collected by sending the robot from one room to the next and measuring the time need to travel a given distance on its path with a stopwatch. The veering of the robot was calculated by sending it a certain distance and measuring the rover's horizontal deviation. We organized and analyzed the data in tables and graphs that were generated by Excel.



Bill of Materials

Quantity	Item	Description	Cost/Unit	Total Cost	Actual Cost
1	DFRobot 4WD Arduino Mobile Platform	HERMES robot, traverses school	$580.00	$580.00	$0.00
3	Arduino Leonardo	first one collects input data from sensors, second one outputs data to motors	$24.95	$74.85	$0.00
1	Sparkfun Monster Moto Shield	Arduino mount, controls DC motors	$79.95	$79.95	$0.00
1	VoomPC-2	runs Master Python Code, data analysis	$79.95	$79.95	$0.00
1	Vex 5" Wheel (4-pack)	supports rover weight, aids in traversal	$19.99	$19.99	$0.00
1	Hokuyo URG-04LX-UG01 Laser Range Finder	detects nearby obstacles	$1,175.00	$1,175.00	$0.00
1	Universal Power Group UB12120 12V Lead-Acid Rechargeable Battery	power supply for motors, Arduinos and computer	$29.95	$29.95	$0.00
1	Lilliput 7" SKD Open Frame Touch Screen VGA Monitor with HDMI, DVI Input	user display, allows user to input destination, tracks rover progress	$164.95	$164.95	$0.00
1	Toys & Juvenile Furniture Police Light & Electric Horn Dome Light	siren to warn nearby people of rover	$8.49	$8.49	$8.49
1	Dell 6ft 18 Pin DVI-D Cable	connects computer to touchscreen	$5.49	$5.49	$0.00
3	Mediabridge Micro USB to USB Cable	connects computer to Arduinos	$5.99	$17.97	$0.00
1	3D Printed Mounts	compass mount, ring for power supply switch	$0.76	$0.76	$0.00
1	0.75" PVC Pipe - 2 feet long	suspends compass above rover	$7.39	$7.39	$0.00
1	OSEPP Compass Sensor	measures heading direction of rover	$37.61	$37.61	$37.61
		Total Project Cost:			$2,282.35
		What We are Paying:			$46.10

Figure 9. Bill of Materials

Figure 10. Gantt Chart

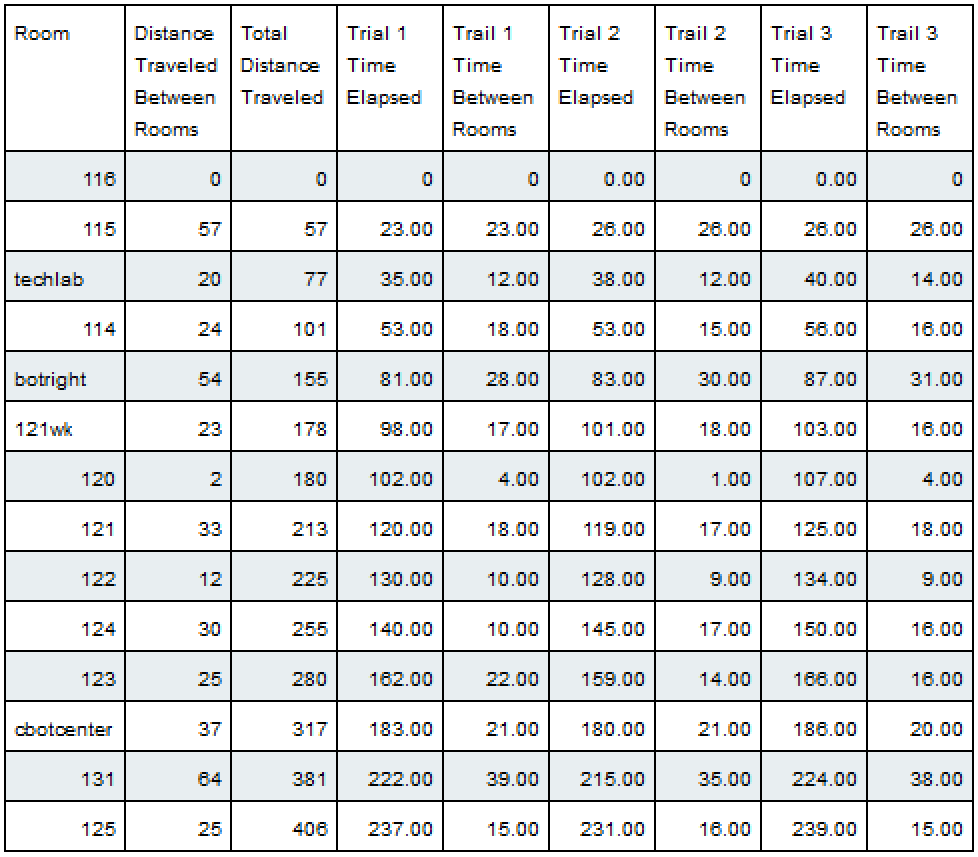
Results

Overall, the robot was able to navigate the school successfully. With the initial and final room, it understood when to move forward and when to turn. Every time that it reached a room, the rover would momentarily stop, signaling that it recognized its location. It could travel between the two furthest points in the school in approximately 7 minutes 20 seconds over a total distance of 754 feet, which met our quantitative goal of traversing the school in under eight minutes. However, an issue was that the robot needed to be continually adjusted by a student during its path; in a distance of 25 feet, the robot moved off course up to 3 feet. In addition, the robot sometimes had trouble navigating turns exactly, and manual adjustment was also needed. The robot had no issues with code but was inhibited by hardware issues.

The compass did not perform as effectively as hoped. Originally, the compass was to be accurate to within one degree, and it would be used to keep the robot on course. The high accuracy requirement was because a small deviation would eventually make the robot go far off course with the distances. However, the compass failed to meet the requirement. The compass would be off by different amounts in each direction, and in some directions the compass was off by about 20 degrees. The issue was probably caused by the quality of the compass and the tilt, since the hallway floor was not the same slope everywhere.

The compass was adjusted by rounding the value to the nearest 90 degree angle (north, south, east, or west). This way, the compass read the general direction so that the general heading would be known, and the direction to turn at intersections could be determined. However, the ability to adjust small deviations to keep the robot straight was lost.

Another problem was the wheels. The original wheels of the DFRobot deformed under the weight of the battery and computer, causing severe veering. New couplets were created and attached to vex wheels with an axle and set screws. The robot initially moved perfectly straight and made almost exact ninety degree turns. As the robot traveled, however, the set screws would loosen after about a minute of running. The veering of the robot would progressively get worse and make turns far off from ninety degrees. The set screws had to be tightened after every run. Thus, each run, the robot would start off fine but could not pass over long distances on its own.



Room	Distance Traveled Between Rooms	Total Distance Traveled	Trial 1 Time Elapsed	Trail 1 Time Between Rooms	Trial 2 Time Elapsed	Trail 2 Time Between Rooms	Trial 3 Time Elapsed	Trail 3 Time Between Rooms
116	0	0	0	0	0.00	0	0.00	0
115	57	57	23.00	23.00	26.00	26.00	26.00	26.00
techlab	20	77	35.00	12.00	38.00	12.00	40.00	14.00
114	24	101	53.00	18.00	53.00	15.00	56.00	16.00
botright	54	155	81.00	28.00	83.00	30.00	87.00	31.00
121wk	23	178	98.00	17.00	101.00	18.00	103.00	16.00
120	2	180	102.00	4.00	102.00	1.00	107.00	4.00
121	33	213	120.00	18.00	119.00	17.00	125.00	18.00
122	12	225	130.00	10.00	128.00	9.00	134.00	9.00
124	30	255	140.00	10.00	145.00	17.00	150.00	16.00
123	25	280	162.00	22.00	159.00	14.00	166.00	16.00
cbotcenter	37	317	183.00	21.00	180.00	21.00	186.00	20.00
131	64	381	222.00	39.00	215.00	35.00	224.00	38.00
125	25	406	237.00	15.00	231.00	16.00	239.00	15.00

Figure 11. Table of times taken to reach rooms starting from room 116

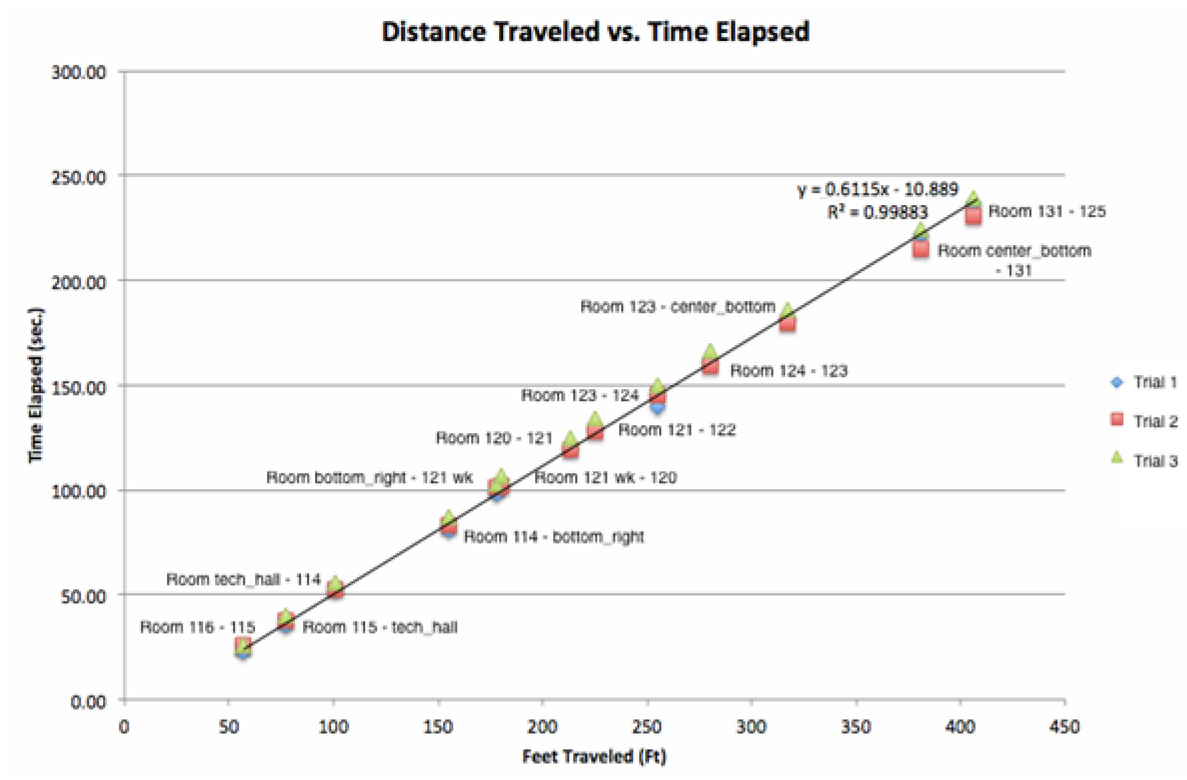


Figure 12. Relationship of time taken versus distance

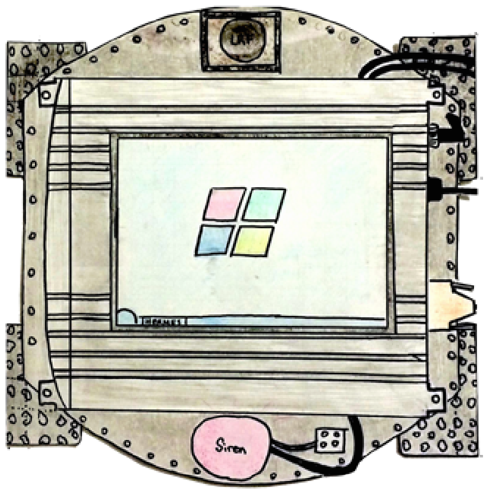


Figure 13. Schematic of robot

Discussion

In terms of code, HERMES was a success because it knew what actions to take when it reached certain points in the hallways. Also, the time requirement of eight minutes to reach anywhere on the first floor was met. However, the robot was only able to navigate through the hallways with constant manual adjustment because of issues with the wheels and compass. With higher quality couplets and wheels and a better compass, the same code would theoretically work perfectly.

Compared to previous research, HERMES had some relative strengths and weaknesses. The rover employed a navigation algorithm mirroring that of the Sojourner Mars rover, as HERMES looked for discrepancies between the local and global maps. Similarly, the Sojourner Mars rover overlapped satellite images with rover-generated topological maps and identified differences between the two. Without use of the Laser Range Finder though, HERMES could not react to real-time changes like the robots in RoboCup or collect data like the Mars Rover. In addition, the hardware issues made movement less precise for HERMES. However, HERMES served a different purpose in that its goal was to find a path indoors. The algorithm is versatile because it can apply to many indoor environments given inputs of a set of coordinates. It can be applied to perform unique functions such as leading a tour or delivering objects.

The results from this study can be used specifically for self localization and navigation within building complexes. The rover can be applied to any global environment given coordinate inputs. Such an algorithm can be used effectively for familiar environments already mapped in detail, but would be hard to apply to explore new environments.

Conclusion

This study showed us the engineering design process and highlighted general trends in research projects. We learned on multiple occasions that engineered products rarely turn out correctly the first time. After assuming that the wheels and compass would function properly from the get-go, we were surprised to find that we needed to make several rounds of modifications in order to achieve the desired results. More specifically for our project, we learned of the importance of data storage, as the transmission and analysis of continuously acquired information demanded a computer with a considerable storage capacity. However, we have yet to fully understand the trigonometric applications behind obstacle avoidance, as the laser range finder outputs an enormous array of data that we cannot appropriately break down and utilize at the current time.

The strengths of the study were concentrated around the programming subsystem. Our incorporation of Dijkstra's algorithm was an eloquent general solution to the navigation problem, and we could theoretically apply our code to any known global environment. This leads us to direct applications of the study, such as a further emphasis on self localization among space exploration rovers akin to those developed by NASA. On the other hand, the weaknesses and shortcomings of the study focused on the mechanical components. Our original wheels could not support the weight of the battery and computer, leading to a whole host of issues and causing us to make several mid-year project changes. We recommend that future researchers of self localization either invest heavily in mechanical platforms that can endure significant stress or enlist the assistance of individuals with welding experience to construct robotic frames.

Final Comments

We would like to thank our lab director, Mr. Charles DelaCuesta, for his support and funding for our project. We would also like to thank Mr. Clinton Behling for allowing us to use the resources in his lab.

Works Cited

DefenceTalk. Retrieved May 4, 2014, from

http://www.defencetalk.com/bomb-disposal-robot-put-to-work-in-afghanistan-23733/.

Gerbis, Nicolas (2012). How the Mars Curiosity Rover Works. *HowStuffWorks*. Retrieved from

http://science.howstuffworks.com/mars-curiosity-rover.htm.

GPS Accuracy. (2014, March 17). *GPS.gov:*. Retrieved May 4, 2014, from

http://www.gps.gov/systems/gps/performance/accuracy/.

UK Ministry of Defense. (2010, January 18). Bomb Disposal Robot Put to Work in Afghanistan.